Journal of Network and Computer Applications 37 (2014) 282–292

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

# Suitability of chaotic iterations schemes using XORshift for security applications

Jacques M. Bahi, Xiaole Fang*, Christophe Guyeux, Qianxue Wang

*University of Franche-Comté, FEMTO-ST Institute, UMR 6174 CNRS, Besançon, France*

ARTICLE INFO

*Article history:*
Received 28 August 2012
Received in revised form
27 February 2013
Accepted 2 March 2013
Available online 14 March 2013

*Keywords:*
Pseudorandom number generators
Chaotic sequences
Statistical tests
Discrete chaotic iterations
Information hiding

ABSTRACT

The design and engineering of original cryptographic solutions is a major concern to provide secure information systems. In a previous study, we have described a generator based on chaotic iterations, which uses the well-known XORshift generator. By doing so, we have improved the statistical performances of XORshift and make it behave chaotically, as defined by Devaney. The speed and security of this former generator have been improved in a second study, to make its usage more relevant in the Internet security context. In this paper, these contributions are summarized and a new version of the generator is introduced. It is based on a new Lookup Table implying a large improvement of speed. A comparison and a security analysis between the XORshift and these three versions of our generator are proposed, and various new statistical results are given. Finally, an application in the information hiding framework is presented, to give an illustrative example of the use of such a generator in the Internet security field.



## 1. Introduction

To use a pseudorandom number generator (PRNG) with a large level of security is it necessary to satisfy the Internet security requirements to support activities as e-Voting, information hiding, and the protection of intellectual property (Bahi and Guyeux, to appear; Liu et al., 2007; Yi and Okamoto, 2012). This level depends on the proof of theoretical properties and results of numerous statistical tests. Many PRNGs, based for instance on linear congruential methods and feedback shift-registers (Knuth, 1998; L'ecuyer, 2008; Blaszczyk and Guinee, 2009), have been proven to be secure, following a probabilistic approach. More recently, several researchers have explored the idea of using chaotic dynamical systems to reinforce the security of these important tools (Falcioni et al., 2005; Cecen et al., 2009; Li et al., 2001). But the number of generators claimed as chaotic, which actually have been proven to be unpredictable (as it is defined in the mathematical theory of chaos) is very small.

This paper extends a study initiated in Bahi et al. (2009), Wang et al. (2010), and Bahi and Guyeux (2010), in which we tried to fill this gap. In Bahi and Guyeux (2010), it is proven that chaotic iterations (CIs), a suitable tool for fast computing iterative algorithms, satisfy the topological chaotic property, as it is defined by Devaney (1989). In Bahi et al. (2009) the chaotic behavior of CIs is exploited in order to obtain an unpredictable PRNG, which depends on two logistic maps. Lastly, in Wang et al. (2010), a new version of this generator using decimations has been proposed and XORshift has replaced the logistic map. We have shown that, in addition to being chaotic, this generator can pass the NIST (National Institute of Standards and Technology of the U.S. Government) battery of tests (NIST Special Publication 800-22 rev1a, 2010), widely considered as a comprehensive and stringent battery of tests for cryptographic applications.

In this paper, a new version of this chaotic PRNG is introduced. It is based on a Lookup Table (LUT) method. After having introduced it, we will give a comparison of the speed, of the statistical properties, and of the security for all of these generators based on XORshift generator (Marsaglia, 2003). These results added to its chaotic properties allow us to consider that this new generator has good pseudorandom characteristics and is able to withstand attacks. After having presented the theoretical framework of the study and a security analysis, we will give a comparison based on new statistical tests. Finally a concrete example of how to use these pseudorandom numbers for information hiding through the Internet is detailed.

The remainder of this paper is organized in the following way. In Section 2, some basic definitions concerning chaotic iterations and PRNGs are recalled. Then, the generator based on LUT discrete chaotic iterations is presented in Section 3. In Section 4, various tests are passed to make a statistical comparison between this new PRNG and other existing ones. In the next sections, a

* Corresponding author. Tel.: +33 381666948.
*E-mail addresses:* jacques.bahi@univ-fcomte.fr (J.M. Bahi), xiaole.fang@univ-fcomte.fr (X. Fang), christophe.guyeux@univ-fcomte.fr (C. Guyeux), qianxue.wang@univ-fcomte.fr (Q. Wang).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.03.001

potential use of this PRNG in some Internet security field is presented, namely in information hiding. The paper ends with a conclusion section where the contribution is summarized and intended future work is presented.

## 2. Review of basics

### 2.1. Notations

| | |
|---|---|
| $[\![1;N]\!]$ | $\rightarrow \{1,2,\ldots,N\}$ |
| $S^n$ | $\rightarrow$ the $n$th term of a sequence $S=(S^1,S^2,\ldots)$ |
| $v_i$ | $\rightarrow$ the $i$th component of a vector: $v=(v_1,v_2,\ldots,v_n)$ |
| $f^k$ | $\rightarrow$ $k$th composition of a function $f$ |
| *strategy* | $\rightarrow$ a sequence which elements belong in $[\![1;N]\!]$ |
| $\mathbb{S}$ | $\rightarrow$ the set of all strategies |
| $\mathbf{C}_n^k$ | $\rightarrow$ the binomial coefficient $\binom{n}{k}=\frac{n!}{k!(n-k)!}$ |
| $\hat{}$ | $\rightarrow$ the bitwise exclusive or |
| $+$ | $\rightarrow$ the integer addition |
| $\ll$ and $\gg$ | $\rightarrow$ the usual shift operators |
| $(\mathcal{X},\mathrm{d})$ | $\rightarrow$ a metric space |
| $\lfloor x \rfloor$ | $\rightarrow$ returns the highest integer smaller than $x$ |
| $n!$ | $\rightarrow$ the factorial $n!=n\times(n-1)\times\cdots\times 1$ |
| $\mathbb{N}^*$ | $\rightarrow$ the set of positive integers {1, 2, 3,…} |
| & | $\rightarrow$ the bitwise AND |

### 2.2. Chaotic iterations

**Definition 1.** The set $\mathbb{B}$ denoting {0,1}, let $f:\mathbb{B}^N\longrightarrow\mathbb{B}^N$ be an "iteration" function and $S\in\mathbb{S}$ be a chaotic strategy. Then, the so-called *chaotic iterations* are defined by Robert (1986)

$$\begin{cases} x^0\in\mathbb{B}^N, \\ \forall n\in\mathbb{N}^*,\ \forall i\in[\![1;N]\!],\ x_i^n=\begin{cases} x_i^{n-1} & \text{if } S^n\neq i \\ f(x^{n-1})_{S^n} & \text{if } S^n=i. \end{cases} \end{cases} \tag{1}$$

In other words, at the $n$th iteration, only the $S^n$-th cell is "iterated". Note that in a more general formulation, $S^n$ can be a subset of components and $f(x^{n-1})_{S^n}$ can be replaced by $f(x^k)_{S^n}$, where $k<n$, describing for example delays transmission. For the general definition of such chaotic iterations, see, e.g., Robert (1986).

Chaotic iterations generate a set of vectors (Boolean vectors in this paper), they are defined by an initial state $x^0$, an iteration function $f$ and a chaotic strategy $S$.

**Algorithm 1.** An arbitrary round of the old CI(XORshift1, XORshift2) generator.

```
a ← XORshift1()
m ← a mod 2 + c
while i = 0,...,m
  b ← XORshift2()
  S ← b mod N
  x_S ← x̄_S
end while
r ← x
Return r
```

### 2.3. Old CI(XORshift, XORshift) algorithm

The basic design procedure of the old CI generator (Bahi et al., 2009) is recalled in Algorithm 1. The internal state is $x$ ($N$ bits), the output state is $r$ ($N$ bits), $a$ and $b$ are computed by two XORshift generators. Finally, $N$ and $c\geq 3N$ are constants defined by the user.

### 2.4. New CI(XORshift, XORshift) algorithm

Algorithm 2 summarizes (Wang et al., 2010) the basic design procedure of the new generator. The internal state is $x$ (a Boolean vector of size $N$), the output state is $r$ ($N$ bits). $a$ and $b$ are those computed by the two XORshifts. The value $f(a)$ is an integer, defined as in Eq. (2). Lastly, $N$ is a constant defined by the user.

$$m^n=f(y^n)=\begin{cases} 0 & \text{if } 0\leq\dfrac{y^n}{2^{32}}<\dfrac{C_N^0}{2^N} \\ 1 & \text{if } \dfrac{C_N^0}{2^N}\leq\dfrac{y^n}{2^{32}}<\sum_{i=0}^{1}\dfrac{C_N^i}{2^N} \\ 2 & \text{if } \sum_{i=0}^{1}\dfrac{C_N^i}{2^N}\leq\dfrac{y^n}{2^{32}}<\sum_{i=0}^{2}\dfrac{C_N^i}{2^N} \\ \vdots & \vdots \\ N & \text{if } \sum_{i=0}^{N-1}\dfrac{C_N^i}{2^N}\leq\dfrac{y^n}{2^{32}}<1. \end{cases} \tag{2}$$

**Algorithm 2.** An arbitrary round of the new CI(XORshift1, XORshift2) generator.

```
1:  while i = 0,...,N do
2:    d_i ← 0
3:  end while
4:  a ← XORshift1()
5:  m ← f(a)
6:  k ← m
7:  while i = 0,...,K do
8:    b ← XORshift2() mod N
9:    S ← b
10:   if d_S = 0 then
11:     x_S ← x̄_S
12:     d_S ← 1
13:   else if d_S = 1 then
14:     k ← k + 1
15:   end if
16: end while
17: r ← x
18: Return r
```

## 3. LUT CI(XORshift, XORshift) algorithms and example

### 3.1. Introduction

The LUT CI generator is an improved version of the new CI generator. The key-ideas are

- To use a Lookup Table for a faster generation of strategies. These strategies satisfy the same property than the ones provided by the decimation process.
- And to use all the bits provided by the two inputted generators (to discard none of them).

These key-ideas are put together by the following way.

Let us firstly recall that in chaotic iterations, only the cells designed by $S^n$-th are "iterated" at the $n$th iteration. $S^n$ can be either a component (*i.e.*, only one cell is updated at each iteration, so $S^n\in[\![1;N]\!]$) or a subset of components (any number of cells can be updated at each iteration, that is, $S^n\subset[\![1;N]\!]$). The first